

Linux Kernel State Tracer (LKST)の 組み向けポーティング と 活用事例

(株)日立製作所

茂岡 知彦

長野 岳彦

目次

LKST概要
ポーティング
適用事例

HITACHI
Inspire the Next

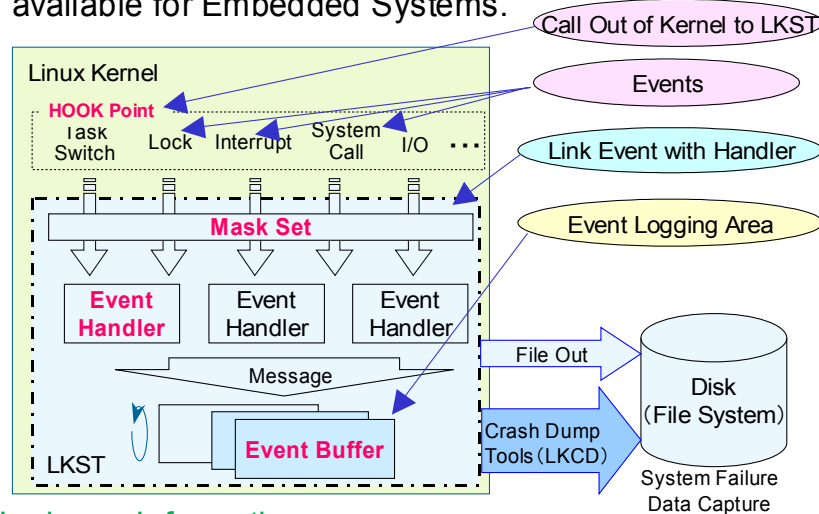


LKST: Linux Kernel State Tracer

Renesas, Hitachi, Lineo

What is demonstrated

LKST is a kernel debugging tool for Linux which traces kernel state transition. LKST was originally implemented on IA-32 based PC server and now is available for Embedded Systems.



Hardware Information

IA-32 PC Server
SH-4(RTS7751R2D), MIPS(RHBMA4400CE),
OMAP(Innovator)

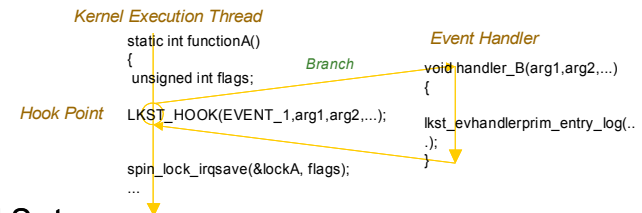
Patch Availability

The patch will be available in the forum patch archive
and at <http://sourceforge.net/projects/lkst/>

How was the Linux improved

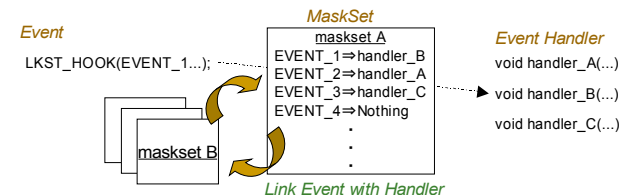
● Hooks

-- fast and slim-line insertion to arbitrary kernel locations



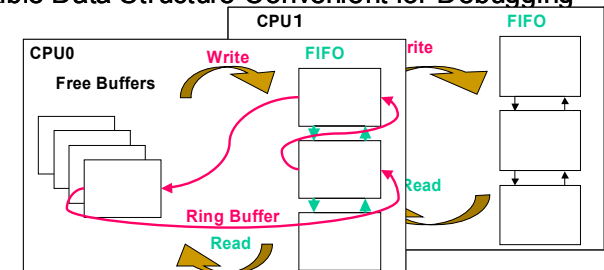
● MaskSet

-- Linking Event with Event Handler



● Event Buffer

-- Flexible Data Structure Convenient for Debugging



LKSTポーティング対象

- カーネルバージョン
 - kernel.org-2.6.9+アーキテクチャパッチ
 - 2.4系(CELinux-040503, MontaVista Linux 3.1)
- アーキテクチャ
 - MIPS32(TX4937)・・・Toshiba RBHMA4400
 - ARM9(OMAP1510)・・・TI OMAP Innovator Kit
 - SH4(SH7751R)・・・Renesas RTS7751R2D

} Hitachi

} Renesas & Lineo
- LKSTバージョン
 - LKST 2.2.1(kernel2.6.9対応)

ポーティングマトリクス

version arch	kernel.org 2.6.9	CELinux 040503	MontaVista 3.1
i386	○		
x86_64	○		
ia64	○		
mips	○	○	○
arm	○		
sh4	○		

- イベント種別
 - アーキテクチャにより異なる機能
 - カーネルバージョンにより異なる機能
- イベントフックポイント実装部
 - アセンブラ記述部
 - アーキテクチャ依存実装部
- タイムスタンプ精度および取得方法

- CPU例外
 - CPUにより例外の種別と発生時取得する情報が異なる
- システムコール
 - i386はシステムコールの入口が2つ、mips/arm/sh4は1つ
- ロック
 - 組込み系でSMPカーネルの利用は稀(TX49はconfig不可)
⇒アーキテクチャ依存スピンロックを省略
(アーキテクチャ共通スピンロックは元々LKSTのフックなし)
- watch dog
 - 86系のみNMIを使用したwatch dogあり
- BHタスクレット起動
 - カーネル2.6からbh_action関数が廃止

- LKST開始

- 現在時刻gettimeofday()①の記録
- タイムスタンプ(machine counter)②の取得と記録
- タイムスタンプ周波数③の取得と記録

LKST初期情報

開始時刻
開始タイムスタンプ
周波数

- イベント発生

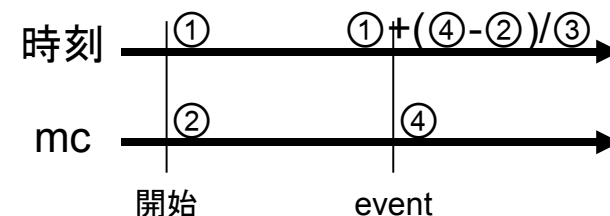
- イベント発生時のタイムスタンプ(machine counter)④の取得と記録

LKSTログバッファ

seq	mc	event	event arg
...
...

- ログ整形出力

- $① + (④ - ②) \div ③$ をasctimeに変換



タイムスタンプの取得

- オリジナル(i386, x86_64, ia64)
 - TSCレジスタの値を参照する⇒86系専用！！
 - TSCレジスタはクロックごとにインクリメントされる64ビットのカウンタ
 - プラットフォーム非依存な方法
 - jiffiesの値を参照する
 - xtimeの値を参照する
 - do_gettimeofday()を呼ぶ
 - MIPS共通の方法
 - CP0(Co-Processor 0)のカウンタレジスタの値
 - TX49固有の方法
 - 内蔵タイマが3ch程度あるので利用可能？
 - OMAP固有の方法
 - 内蔵タイマが3ch程度あるので利用可能？
 - SH固有の方法
 - OMAP同様内蔵タイマがあるので利用可能？
- kernelのconfig
で選択
- 固有実装を入れる
口だけ用意。
実装はご自由に
。
- kernelのconfig
で選択

タイムスタンプ各種の比較

方法	詳細	注意点
jiffies	カーネル内変数jiffiesを参照する。	桁、精度はHZ次第(2.6.9-TX49は1000) 2.6.9では64bit整数で、直接参照禁止(下32bitはOK)
xtime	カーネル内変数struct timespec xtimeを参照する。 <pre>struct timespec { time_t tv_sec; long tv_nsec; };</pre>	排他必要で直接参照できず、関数呼び出し current_kernel_time() 桁は多い(nsec)が精度はハードにより違う
gettimeofday	do_gettimeofday()を呼び出す。	処理が複雑で重い。 桁は多い(μsec)が精度はハードにより違う
MIPS-CP0	CP0のカウンタレジスタを参照する。	精度はCPU周波数の1/2。 レジスタの幅が32bit

- jiffies
 - 2.6.9では初期値が $-300 \times \text{HZ}$ なので時間が戻る！
⇒常にjiffiesに $300 \times \text{HZ}$ を足す
 - LKSTはTSCの周波数が 1KHz 以上を前提
⇒jiffiesの値をmsec単位に変換
- MIPS CP0カウンタレジスタ
 - 32bitなのですぐ1周する(300MHz で28秒)
⇒タイマ割込み処理でオーバー回数をカウント
 - 整形出力するとイベント発生時刻がだんだんずれる
⇒カーネルの認識したクロック周波数が間違っていたので修正

実は、カーネルのタイマ(ハイレゾタイマ)関係の部分に
似た処理が存在した...

- 新規アーキテクチャサポートに必要な作業
 - 一部コア機能のプラットフォーム固有実装
 - arch/HOGE include/asm-HOGE/以下のイベントフックのうめこみ
- コア機能(必須)
 - machine counterとその周波数の取得
 - program counter取得
 - ログレコード番号のアトミックインクリメント
- arch/HOGEのイベントフック埋め込み(省略可能)
 - システムコールエントリ(scall32-o.S, entry.Sなど)
 - トラップや例外処理(traps.c)
 - 割り込み(irq.c)
 - メモリ管理の一部(fault.c)
 - カーネルスレッド生成(process.c)
 - I/O(io.h)
 - アーキテクチャ依存スピンロック(spinlock.h)
 - 埋め込んだイベントの一覧(lkst_etypes.h)

- カーネル2.4系へのバックポート
 - 状況
 - 本家LKSTではカーネル2.4のサポートはすでに廃止
 - Miracle Linux V3(86系)が2.4ベースでLKST 2.2.1をサポート
 - 作業
 - Miracleを参考にLKSTをCELinuxとMontaVistaにとりこむ
 - mips対応部分をとりにこみ

- ハードウェア割込みの頻度を測定
 - 問題：ある製品用にLinuxを移植作業中、異常なCPU負荷が長時間連続する問題が発生。
 - CPU利用率をvmstatで調査したところ、割込みが多発している事が判明
 - /proc/interrupts以下のデータを、定期的に出力して、H/Wデバイスからの割込み状況をチェック
 - I2Cバス用のドライバが異常な負荷を発生している。
 - LKSTを使用して、IRQ番号毎の割込みの状況を正確にチェックし、開発者側の考えの妥当性を検証した。

- LKSTの使用方法
 - ハードウェア割込みのイベントのみをトレース
 - 記録するイベントをINT_HARDWARE_ENTRYのみに設定。
- LKSTのトレース結果をスクリプトで解析
 - LKSTの出力結果より、必要な情報のみをCSV形式に変換

```
lkstbuf print -rCSV -f $1 | cut -d, -f 1,2,3,4,5,8,11,14,17  
| tr ',' ' ' > trace.csv
```

"interrupt hardware_entry"	00	00000206	946685659 119999000
----------------------------	----	----------	---------------------

イベント名称

プロセスID

イベント発生時刻

0x00000025	0x00000001	0x80f99f34	0x00000000
------------	------------	------------	------------

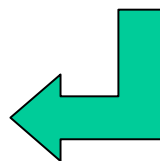
IRQ番号

適用事例 割り込み頻度の測定（３）

- CSV形式に整形したデータより、発生したirq番号を整理

- `grep interrupt hardware_entry $TRACELOG | cut -d ' ' -f 7 | sort | uniq > $IRQDB`

ユニークなIRQ番号を取得



```
$IRQDB
0x0000000e
0x00000010
~
0x00000024
0x00000025
```

- `grep interrupt hardware_entry $TRACELOG | grep -c $IRQNO`
IRQDBのIRQナンバ毎に、イベントの発生回数をカウント

• 調査結果の例

IRQ noより、I2Cバスからの
割り込みである事がわかる。

=====

```
total >> 23312
```

=====

```
IRQ_NO = 0x00000000, count = 1998
IRQ_NO = 0x00000001, count = 923
IRQ_NO = 0x00000001, count = 41
IRQ_NO = 0x0000001b, count = 11887
IRQ_NO = 0x0000001c, count = 1
IRQ_NO = 0x0000001d, count = 799
IRQ_NO = 0x0000001e, count = 138
IRQ_NO = 0x00000022, count = 1344
IRQ_NO = 0x00000023, count = 910
IRQ_NO = 0x00000024, count = 587
IRQ_NO = 0x00000025, count = 4684
```

=====

- カーネルランドのセマフォのロック/アンロックが正しく動作しているか確認
 - 問題：移植中のファイルシステムにおいて、書き込み処理が失敗する。
 - 問題のファイルシステムは、down_trylock関数の処理中に停止する。
 - カーネルランドのセマフォの取得/開放が正しく実施されているか
 - LKSTを使用して、カーネルランドのセマフォが正常に動作しているかチェックした。

- LKSTには、カーネルセマフォのフックポイント
は無いので、新規にフックを追加した。
 - カーネル MontaVista Linux 3.1
 - 変更ファイル arch/mips/kernel/semaphore.c
 - 変更関数 __up(), __down_trylock()

・カーネルセマフォのHook追加例

```
void __up(struct semaphore *sem)
{
    wake_one_more(sem);
    wake_up(&sem->wait);
    //----- added -----[start]-----
    LKST_HOOK(LKST_ETYPE_K_SEM_UP,
    LKST_ARG(&sem->count),
    LKST_ARG(&sem->sleepers),
    LKST_ARG(0),
    LKST_ARG(0));
    //----- added -----[ end ]-----
}
```

```
void __down_trylock(struct semaphore * sem)
{
    //-----added -----[start]-----
    LKST_HOOK(LKST_ETYPE_K_SEM_DOWN,
    LKST_ARG(&sem->count),
    LKST_ARG(&sem->sleepers),
    LKST_ARG(0),
    LKST_ARG(0));
    //-----added -----[ end ]-----
    return waking_non_zero_trylock(sem);
}
```

• lkst_etype.hの修正

//0x106 event
LKST_ETYPE_DEF(, NORMAL, , "kernel semaphore is up", \
 "sem_count", \
 "sem_sleepers", \
 "NULL", \
 "NULL")

イベント番号 イベント名

//0x107 event
LKST_ETYPE_DEF(0x107, NORMAL, K_SEM_DOWN, "kernel semaphore is down", \
 "sem_count", \
 "sem_sleepers", \
 "NULL", \
 "NULL")

- 結果例

event_type=kernel semaphore is up
cpu=00, pid=00000129
time=Sat Jan 01 00:04:16.245140414 2000
arg1=0x00000000 00000000 : sem_count
arg2=0x00000000 00000000 : sem_sleepers
arg3=0x00000000 00000000 : NULL
arg4=0x00000000 00000000 : NULL